REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31-08-2005	FINAL	Sept 9, 2004 – March 8, 2005

4. TITLE AND SUBTITLE

COLLABORATIVE SOFTWARE FOR INFORMATION FUSION

5a. CONTRACT NUMBER
DAAD19-01-C-0065

5b. GRANT NUMBER
8005.039.46

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. Daniel D. Corkill

5d. PROJECT NUMBER

5e. TASK NUMBER
39

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Massachusetts Amherst
Department of Computer Science
140 Governors Dr., CMPSCI Bldg Rm 100
Amherst MA 01003

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Micro Analysis and Design, Inc.
4949 Pearl East Circle, Suite 300
Boulder CO 80301

10. SPONSOR/MONITOR'S ACRONYM(S)
ARL MAAD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

This report describes initial research in developing the scientific foundations and practical experience necessary for a highly responsive information-fusion application that improves the effectiveness of analysts and decisionmakers within the Army's Unit of Action (brigade-level force). This research is leading to the development of a software application that can augment and support Army personnel in answering Priority Intelligence Requirements (PIRs) associated with monitoring, assessing, and responding to enemy actions and other battlespace-environment characteristics. Currently, time constraints and information overload often result in hasty, partial analysis of the information available to intelligence personnel. An effective, automated support application can help Army analysts and decisionmakers within the Unit of Action focus on appropriate data by providing spatially and temporally aggregated views of the environment and by ensuring that important information has not been overlooked. Initial research was performed in the areas of: blackboard-system-based architectural techniques, opportunistic control machinery, and their effects on hypothesis management; multi-entity Bayesian blackboard representations, construction, and inference; temporal and spatial knowledge representation and data aggregation; dynamic, priority-based, problem-solving control strategies. This report discusses issues and approaches addressed, progress to date, and lessons learned—concluding with a summary of technical challenges and recommendations facing future research and development activities.

15. SUBJECT TERMS
Multi-level fusion; PIR analysis; decision support; blackboard architecture; spatial and temporal aggregation; opportunistic control

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Daniel D. Corkill
Unclassified	Unclassified	Unclassified	Unclassified, unlimited	32	19b. TELEPHONE NUMBER *(include area code)* 413-545-0675

Collaborative Software for Information Fusion

Final Report

Period of Performance: September 9, 2004–March 8, 2005
Sponsored by: U.S. Army Research Laboratory
U.S. Army RDECOM CERDEC I2WD
Contract: DAAD19-01-C-0065
Prime: Micro Analysis and Design, Inc.
Agreement Number: 8005.039.46 Task #39

Daniel D. Corkill

Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
corkill@cs.umass.edu

March 2005

Abstract

This report describes initial research in developing the scientific foundations and practical experience necessary to create a highly responsive information-fusion application that improves the effectiveness of analysts and decision makers within the Army's Unit of Action (brigade-level force). This research is leading to the development of a software application that can augment and support Army personnel in answering Priority Intelligence Requirements (PIRs) associated with monitoring, assessing, and responding to enemy courses of action and other battlespace-environment characteristics. At present, time constraints and information overload often result in hasty, partial analysis of the information available to intelligence personnel. An effective, automated support application can help Army analysts and decision makers within the Unit of Action focus their attention on appropriate data by providing spatially and temporally aggregated views of the environment and by ensuring that important information has not been overlooked. Initial research activities were performed in the areas of: 1) blackboard-system-based architectural techniques, opportunistic control machinery, and their effects on hypothesis management; 2) multi-entity Bayesian blackboard representations, construction, and inference; 3) temporal and spatial knowledge representation and data aggregation; and 4) dynamic, priority-based, problem-solving control strategies. This report discusses the issues and approaches we addressed, our progress to date, and lessons learned. The report concludes with a summary of remaining technical challenges and recommendations for future research and development activities.

The research reported in this document was funded by the Fusion Based Knowledge for the Future Force Program led by the Intelligence and Information Warfare Directorate of the U.S. Army RDECOM CERDEC at Fort Monmouth, NJ. It was performed in connection with contract DAAD19-01-C-0065 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the author and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, the University of Massachusetts, Micro Analysis and Design, Inc., or the U.S. government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Contents

1 Objective

The overall objective of this research effort is developing the scientific foundation and experience necessary to create a highly responsive information-fusion application that improves the effectiveness of Army analysts and decision makers within the Army's Unit of Action (brigade-level force). These analysts and decision makers must work with large data volumes in time-constrained and uncertain operating environments.

The specific context of this research is a task-oriented, land-based battlespace data-fusion Army Technology Objective (ATO-Research) program entitled Fusion Based Knowledge for the Future Force (FBKFF). The Intelligence and Information Warfare Directorate of CERDEC, Fort Monmouth, NJ is the ATO Manager of FBKFF. FBKFF is designed to augment and support field personnel in answering Priority Intelligence Requirements (PIRs) associated with monitoring, assessing, and responding to enemy courses of action and other battlespace-environment characteristics pertaining to states of the enemy, entities whose allegiance is unknown, non-combatants, terrain, and weather. A major challenge in FBKFF is managing the combinatorial explosion of sensing and processing activities without sacrificing accurate inference. The large volumes of data, possibilities, and outcomes exceed human perceptual and cognitive abilities and require an effective human/computer partnership to make the best use of sensing, computation, and communication resources in highly dynamic and uncertain battlefield environments. At present, time constraints and information overload often result in hasty, partial analysis of the information available to intelligence personnel. An effective, automated support application for information fusion and situation assessment can help Army analysts and decision makers within the Unit of Action focus their attention on appropriate data by providing spatially and temporally aggregated views of the environment and by ensuring that important information has not been overlooked.

The specific objective for this initial "proof of concept" effort was to conduct research and development in the use of advanced collaborative blackboard-system architectures for multi-entity Bayesian model construction and inference. In particular, this research focused on FBKFF-specific representation strategies, control techniques, and their effects on hypothesis management. The large data volumes challenge automated techniques and require an agile, knowledge-intensive architecture that can quickly focus attention on appropriate data and can extract high-level behavioral information that is present in the data. As will be discussed, it quickly became apparent that significant collective information is present in the stream of individual manual and automated intelligence, surveillance, and reconnaissance (ISR) reports that are available to analysts and decision makers. Harvesting this rich collective information requires semantically aggregating individual reports in both space and time. Rather than discarding detailed spatial and temporal information in order to simplify automated reasoning, we began investigating how to make use of all the information that can be obtained from ISR sources and how to automate the semantic aggregation and reasoning that is now performed manually. We also began work on developing control strategies that prioritize activities and information reports in light of current PIRs and the operational situation. Finally, we started to investigate meta-level reasoning that can identify where additional information, if available, would significantly increase confidence in specific answers or greatly reduce the computational effort required in developing an answer. Such power of information analysis can be used to reallocate sensing resources to better achieve operational objectives.

2 Technical Approach

This effort focused on the combined use of advanced collaborative blackboard-system technologies and multi-entity Bayesian model construction and inference. In the FBKFF setting there are many different types of sensors and sensing capabilities that need to be allocated and tasked dynamically to best recognize enemy objectives and courses of action. Similarly, the fusing of reports coming from heterogeneous sensors, geographically distant sensors, and human-generated reports must be adjusted in real-time to focus quickly on the information that is most relevant to current PIRs. These requirements are well suited to the collaborating-software capabilities provided by blackboard-system and MAS technologies.

In addition to real-time, operational agility, the design of the FBKFF application should allow the software approaches and algorithms to be easily changed, improved, and extended throughout the lifetime of the information-fusion application. We should expect from the outset that new and improved sensor types, operational models, software techniques and components will be developed over time and added to the application. The underlying architecture of the application should be able to adapt to the complexity of new capabilities and be able to manage them as part of its ongoing operations. Again, a collaborating-software architecture is ideal for meeting this requirement.

In this effort, we restricted our work to a centralized information-fusion application. Although the sensors and human observers are geographically distributed, all reports are provided to a single location. This eliminated complexities of distributed problem solving from our work and allowed us to direct our attention to issues of combining diverse information, knowledge, and processing techniques within a single computer setting.

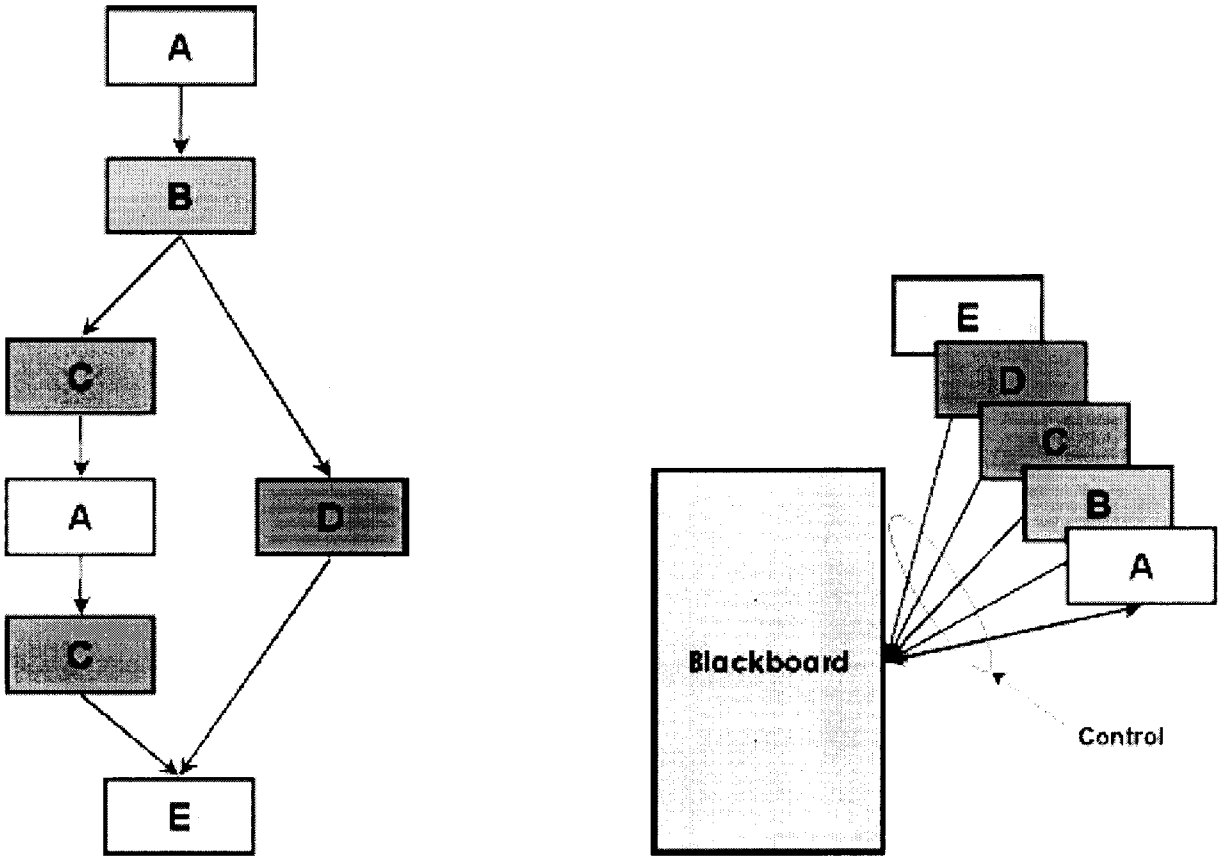
Our work in this effort was roughly divided into two halves. The first half, termed "architectural development," involved constructing the overall application architecture and control machinery, ingesting automated sensor and human reports, and performing initial semantic processing of the report data. The second half, termed "incremental Bayesian reasoning," investigated the use of multi-entity Bayesian representation and reasoning techniques on the objects produced by initial semantic processing. Dr. Daniel Corkill, the PI on this effort, performed the architectural development portion of the work and Gary King worked on the higher-level Bayesian reasoning and preliminary knowledge-engineering portion. Each portion will be discussed in Section 3, but first, we present the technical background of our approach.

2.1 Blackboard Systems

Blackboard-system technologies form the foundation of the information-fusion application. Blackboard systems were the first artificial intelligence (AI) applications involving collaborating software modules [1, 2, 3, 4].[1] The goal in these applications was to achieve the flexible, brainstorming style of problem solving exhibited by a group of diverse human experts working together to address problems that no single expert could solve alone.

A traditional, interface-oriented, way of combining a set of diverse software modules is to connect them according to their data-flow requirements (such as the five modules shown in Figure 1a). When needed, the same modules can appear multiple times in the communication graph, but in this architectural approach the connections are all predetermined and direct. This "hardwired collaboration" approach can work well when both the module set and the appropriate communications among modules do not change. This approach has the advantage of a simple, predictable processing structure and processes that are relatively fixed and understandable. When the specific modules are subject to change and/or when the ordering of modules cannot be determined until specific data values become known at execution time, the inflexibility of direct interaction becomes unwieldy. From a system-building perspective, direct

[1] In the field of software architectures, basic blackboard-style systems are called *repositories* [5, 6].



(a) Directly Connected Modules (b) Anonymously Interacting Modules

Figure 1: Connecting Modules Together

interaction promotes the use of private communication protocols between modules. Such specialized protocols can be made succinct and efficient, but changes to the communication graph or the addition of a new module can require changes to a number of individual communication protocols. Finally, the fixed structure offers limited flexibility in making situation-based control decisions.

Blackboard systems use a contrasting approach where module interaction is indirect and anonymous via an intermediary—in this case, a blackboard data repository (Figure 1b). In the blackboard-system approach, all processing paths are possible, and the choice among paths can be made by a "moderator" mechanism that selects among the possible paths dynamically. The information placed on the blackboard is public and available to all modules, control mechanisms, newly added modules, and monitoring and debugging tools. Indirection significantly reduces the number of communication interfaces that must be supported among highly collaborating modules.

Blackboard systems were developed in the 1970s to avoid the limitations of directly connected architectures, and were first used to solve complex signal-interpretation problems in systems such as Hearsay-II [7, 1] and, shortly thereafter, HASP/SIAP [8]. A blackboard system consists of three main components: knowledge sources (KSs), the blackboard, and a control component. Each of these components contributes to blackboard-system collaborative capabilities.

KSs In a blackboard system, each KS is a specialist at solving certain aspects of the overall application and, in theory, is developed independently of all other KSs. A KS does not need the expertise of other KSs in order to function; it does not even need to be aware of what other KSs might be present in the system. However, it must be able to understand the representation and semantics of relevant

information placed on the blackboard. Each KS also needs to know the conditions under which it can contribute to a solution. This knowledge is called the triggering condition of the KS.

KSs are not the active "agents" in a blackboard system. Instead, KS *activations* are the active entities competing for a chance to make contributions. A KS activation is the combination of the KS knowledge and a specific triggering context. The distinction between KSs and KS activations is important in applications where numerous events occur that trigger the same KS. In such cases, control decisions involve choosing among particular applications of the same KS knowledge (focusing on the appropriate data context), rather than among different KSs (focusing on the appropriate knowledge to apply). KS-activation "agents" remain alive only until the KS activation is executed or is canceled prior to execution.

The Blackboard The blackboard is a shared data structure that is available to all KSs and serves as: 1) a community memory of data, contributions, developing solutions, and control information; 2) a communication medium and buffer, and 3) a KS trigger mechanism. Blackboard applications tend to have elaborate blackboard structures, with multiple representations and levels of abstraction. Although this blackboard organization is useful to the developers and users of the system, the main reason for structuring the blackboard is for efficiency. If a large number of contributions are placed on the blackboard, quickly locating pertinent information becomes a problem. A KS execution (short for the execution of a KS activation) should not have to scan the entire blackboard to see if appropriate items have been placed on the blackboard by another KS execution. The blackboard can be subdivided into regions, levels, planes, or multiple blackboards, each corresponding to particular kinds of information. Additionally, ordering metrics can be used within each region. Advanced blackboard-system frameworks provide rich positional metrics for efficiently locating blackboard objects of interest [9].

An important characteristic of the blackboard approach is the ability to integrate contributions whose *relationships are not known in advance.* For example, a KS execution working on one aspect of a problem may put a contribution on the blackboard that does not seem relevant or interesting to any other KS. Only until much later, when substantial work on other aspects of the problem has been performed, is there enough context for other KSs to recognize the relevance of the early contribution. By retaining these contributions on the blackboard, the blackboard system can remember these early problem-solving efforts, avoiding the need to recompute them later (when their importance is understood) or losing them altogether (if no entity chose to keep them around). Additionally, the blackboard control component can notice when highly promising contributions placed on the blackboard remain unused by other KS executions and possibly choose to apply some problem-solving resources on understanding why they did not fit with other contributions. Many contributions placed on the blackboard may never prove useful. Therefore, a KS execution must be able to efficiently inspect the blackboard to see if relevant information is present [10, 11].[2] This search for relevant information involves: 1) computing approximate attribute values for the kinds of blackboard objects that are relevant given specific KS triggers, and then 2) finding those objects on the blackboard (Figure 2).

The importance of such proximity-based *associative retrieval* to locate relevant objects that have been placed on the blackboard by other KS executions is often overlooked in casual discussions of blackboard systems. Additionally, objects on the blackboard often have significant *latency* between the time they are placed on the blackboard and the time they are determined to be relevant for use by another KS. If it were not for this latency between creation and use, the blackboard in a system with a fixed set of KSs could be replaced with direct calls among the KSs by a configuration-time compiler, and we would be back to the directly connected modules of Figure 1a. It is the temporal separation of the placement of contributions on the blackboard and their possible use that provides blackboard systems with significant flexibility in ordering KS executions. In order to obtain this same flexibility without a shared blackboard, each KS module would have to maintain its own copy of objects received from other modules. Furthermore,

[2]Distributed-object and tuple-based systems, such as JavaSpaces [12], TSpaces [13], or Linda [14], are sometimes suggested as underlying support for blackboard applications. Their performance limitations, especially in supporting rapid, proximity-based search, make them a poor choice.

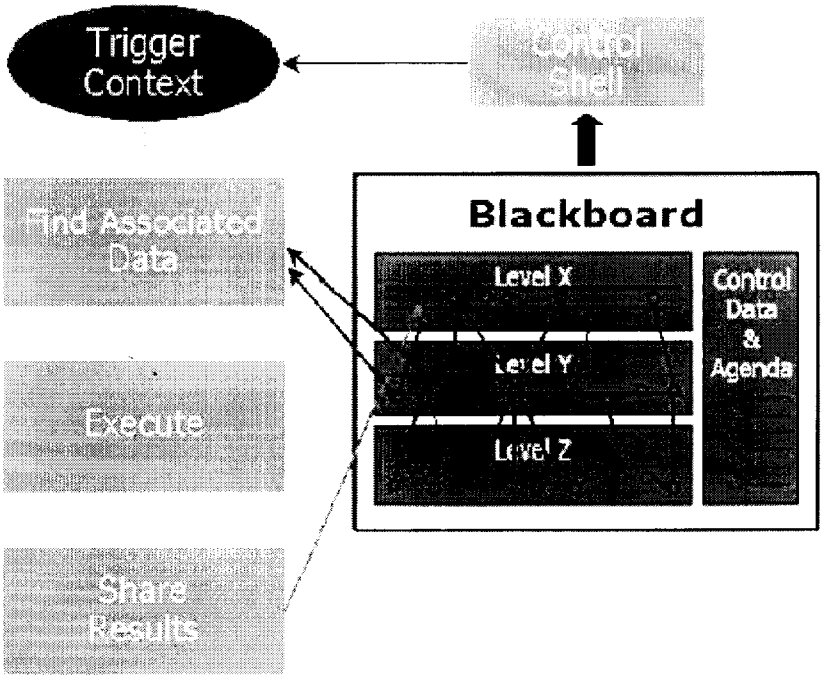


Figure 2: KS Execution Activities

whether the memory is globally shared (on the blackboard) or private (within a KS), an efficient means of locating previously created objects is required.

Latency and proximity-based retrieval also play major roles in swarms and other stigmergetic biological societies, where the physical location of materials left by other entities is a key part of their collaborative behavior. In relating blackboard systems to stigmergetic biological mechanisms, it has been pointed out that "the environment is nature's blackboard" [15]. This common theme is also reflected in recent work on formal computational models for open systems in which latency and observability/ retrieval aspects of anonymous and indirect interaction are being represented [16].

Control Component In a blackboard system, a separate control mechanism, sometimes called the control shell, directs the problem-solving process by managing how KSs respond to contributions placed on the blackboard by an executing KS activation. These contributions trigger events that are maintained (and possibly ranked) by the control shell until the executing KS activation is completed. At that point, the control shell uses the events to trigger and potentially activate KSs. The new KS activations are ranked, and the most appropriate KS activation (old or new) is selected for execution. This KS execution cycle continues indefinitely (for continuous applications) or until the problem is solved (for single-solution-based applications).

The control shell needs to choose pending KS activations without possessing the detailed expertise of all the individual KSs. Without such a separation, the modularity and independence of KSs would be lost. If specific knowledge of all the KSs had to be included within the control shell, it would have to be modified every time a KS was added or removed from the system. On the other hand, control decisions in a blackboard system are to be made by the control shell—not by KSs.

The solution is to separate control knowledge into generic, overall control knowledge contained in the control shell and detailed KS-specific control knowledge packaged with each KS. Then, whenever the control shell needs KS-specific control information, it asks the KS for estimates on how it will behave (Figure 3).

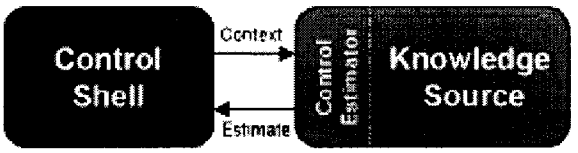


Figure 3: Separation of Control Knowledge

When a KS is triggered, the control shell passes the triggering context to the KS, which uses its KS-specific control knowledge to estimate factors such as the quality, importance, cost, and likelihood of successfully making potential contributions [17]. This estimate is determined without actually performing the work to compute the contributions. Instead, each KS generates estimates of the contributions that would be generated by using fast, low-cost, approximations developed by the KS writer. These estimates are of the form, "If this activation is selected for execution in this context, I estimate it will generate contributions of this type, with these qualities, while expending these resources." The KS returns these estimates to the control shell that uses them in deciding how to proceed.

Large data volumes and the many inferences that can potentially be made from them require that a responsive and effective information-fusion environment focus its processing activities on exploring appropriate inferences using the most appropriate data. Choosing among the myriad of potential processing activities should also be sensitive to the current environmental context and PIRs. Potential processing activities may be contributing to a single line of reasoning, exploring multiple, related lines of reasoning, or working on completely independent information-fusion tasks. Whatever their purpose, the activities are interdependent in their contention for shared resources (such as processing, memory, and communication) and their potential for distracting other software entities from more appropriate activities. Cost and benefit estimates can be used to understand the effect that obtaining particular observations or performing particular processing activities will have on the current assessment of the environment. Such power-of-information and power-of-reasoning control decisions are very useful in making effective use of sensing and processing resources.

Research on blackboard-system, and subsequently on multi-agent system, coordination techniques has been underway for over thirty years starting with early work on Hearsay-II [17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Our control research, and much of the work of others, has focused on developing effective control mechanisms for specific architectural settings and application characteristics.

2.2 Incremental, Mixed-Entity Bayesian Reasoning

The extensive use of graphical models, such as Bayesian networks, in AI applications [27, 28, 29, 30, 31] has led to criticism of the ad hoc confidence and belief values used in traditional blackboard applications. These ad hoc belief values were involved in everything from making control decisions to determining solutions and the system's confidence in them. This criticism has generated considerable interest in developing Bayesian blackboard systems. The idea is to replace ad hoc representations of the relationships among blackboard objects with incrementally generated graphical models. Recent techniques in constructing belief networks using network fragments [32, 33] and in hierarchical object-oriented Bayesian networks [34, 35] have been suggested as candidate technologies that can be extended to create more principled blackboard reasoning.

As part of this effort, we sought to apply some incremental Bayesian information-fusion research done in the Experimental Knowledge Systems Laboratory (EKSL) at UMass Amherst to the FBKFF application. AIID (Architecture for Interpretation of Intelligence Data) is a prototype Bayesian blackboard system for intelligence analysis in which KSs create and manipulate Bayesian graphical models on the blackboard [36, 37]. The AIID prototype was developed by EKSL to experiment with principled blackboard-system inference and control reasoning, but the software was not intended to be used in an application setting. The research question for this effort was exploring the applicability of the AIID approach to the KS reasoning required in FBKFF and, in particular, the ability to scale the AIID approach to large-data-volume settings.

AIID's current beliefs are represented on the blackboard as a possibly disconnected graphical network that includes previous observations, background knowledge, and hypotheses about the data. In the military domain, the blackboard contains nodes that include sightings and hypothesized locations of

enemy units, locations of key terrain, and hypotheses about the enemy's tactics and strategy. AIID uses a common first-order extension to belief networks to represent multiple similar entities more compactly [38]. This approach is analogous to the extension of propositional logic (where atomic formulas are propositional variables) to predicate logic (which is quantificational, where atomic formulas are propositional functions) [39, 40]. Instead of representing random variables by a single name, each node has a node-type and a set of arguments. Logic variables can then be used as arguments in KSs to describe a relationship that does not depend on the particular argument values. The combination of a node-type and arguments uniquely specifies a node on the blackboard.

Time complicates graphical-network representations, and information on AIID's blackboard can occur on significantly different temporal scales. AIID uses two temporal representations: a discrete, tick-based representation and a temporal-interval representation. At lower levels of the blackboard, each network node is indexed by the time it occurs.[3] At higher levels of the blackboard, which contain longer-term actions and intentions, every node is represented by the interval in which it occurs. Each node has a start time and an end time that is also explicitly represented as nodes in the network.

As with propositional logics, complex spatial representation and reasoning are highly problematic for graphical-network approaches. AIID uses procedural KSs to perform geometric reasoning, and the results of such reasoning are then represented using a Bayesian network fragment. This is consistent with the Bayesian blackboard concept where it is the relationships among blackboard objects that are represented using incrementally generated graphical models rather than requiring all reasoning to be performed using Bayesian logic.[4]

These concessions to complexity, as well as other "principled integration" concerns that we will describe, demonstrate that simply using a Bayesian graphical-network representation of blackboard objects does not automatically realize the principled problem solving sought by Bayesian-blackboard proponents. In fact, the emphasis on developing a principled blackboard representation of the developing solution is misplaced. Approaches such as AIID do not explicitly represent the uncertainty/error characteristics of the KSs that build and modify the graphical network. These contribution characteristics are simply rolled into the uncertainly associated with the solution (and essentially ignored). Instead of feeling "principled" about using a formal blackboard representation, the emphasis should be on making the integration of the contributions generated by diverse KS entities, as well as the semantics of the inputs to the system, well founded. This can only be achieved by modeling how these contributions are generated and how they relate to one another. Bayesian techniques remain applicable for representing how the FBKFF system combines the contributions of numerous KSs, but representing how the activities of each KS relate to one another is the key issue—not merely the use of a Bayesian representation on the blackboard.

3 System Architecture and Status

The core architecture for the prototype information-fusion application developed in this effort was implemented using the open-source GBBopen framework.[5] GBBopen is a modern, high-performance, open source blackboard-system development environment that is based on the concepts that were explored and refined in the UMass Generic Blackboard system [42] and the commercial GBB product [43]. GBBopen is not, however, a clone or updated version of either system. The GBBopen Project is applying the knowledge and experience gained with these earlier tools to create a new generation of

[3]Making the graphical network at these levels a Dynamic Bayesian Network [41]

[4]Otherwise, a Bayesian blackboard system would be equivalent to any Bayesian reasoner.

[5]http://GBBopen.org

blackboard-system capabilities and make them freely available to a wide audience.

The GBBopen software provides a number of important benefits:

- A modular, open-source reference implementation of blackboard-system infrastructure that serves as a basis for research and development activities.
- Deployment of robust and high-quality software releases that are validated through a process of widespread peer review.
- Source code and the right to modify it enable unlimited improvement and enhancement of the software. It also makes it possible to port the code to new hardware and software, to adapt it to changing conditions, and to reach a detailed understanding of how GBBopen works. Source-code availability also makes it much easier to isolate and fix bugs.
- The right to redistribute improvements and extensions to the GBBopen source code encourages such developments and enables them to be shared by the user community.
- The right to use the software, combined with redistribution rights, attracts users and sponsors, which encourages further support and extension of the software.
- There is no single entity on which the future of the GBBopen software depends. This is particularly important given the highly specialized nature of blackboard-system software and the lack of multiple implementations. With open-source software, it is always possible for another group to continue maintenance and improvement of the GBBopen software.
- Open-source software enables forking: the creation of an alternative version when development is perceived as not moving in the right direction or quickly enough to meet particular needs. Although forking can be a disadvantage, it allows the concurrent exploration of different approaches by different groups in the development of complex software systems. The modular nature of GBBopen is intended to encourage the creation of alternative and additional GBBopen modules (called "module forks") for use in research and experimentation.

Using the GBBopen software in this project enabled us to incorporate the latest blackboard-system technologies, to avoid re-implementing high quality blackboard-system capabilities, and to focus immediately on the FBKFF-based challenges of the research. GBBopen is written in Common Lisp and utilizes advanced capabilities provided by the Common Lisp Object System Metaobject Protocol (MOP). It has been ported to the following open-source and commercial Common Lisp implementations: Allegro Common Lisp, CLISP, CMUCL, Lispworks, MCL, OpenMCL, and SBCL. A wide range of popular hardware and operating systems are supported by these Common Lisp implementations, and transitively, by the prototype information-fusion application.

Although GBBopen development is ongoing, the Version 0.8 release provided what was needed to support this effort. The biggest limitation to future large-scale work is GBBopen's current lack of optimized set and series composite objects and retrieval mechanisms (work on these is underway) and hashing storage for enumerated dimensions (another performance enhancer).[6] Completion of series-composite optimizations is an important enabler for large-scale experimental work in temporal- and spatial-aggregation reasoning.

3.1 System Inputs

Input to the FBKFF information-fusion application consists of a stream of individual human and automated sensor reports, knowledge of sensor and target types and capabilities, activity and behavioral (doctrinal) and strategic knowledge, terrain and weather features, and so on. Some of the knowledge

[6]The representation and pattern portions of GBBopen composites are in place, but storage and pattern optimizations to obtain top performance with large composite data volumes are not finished. In this effort, the lack of these optimizations was not a major hindrance.

is represented declaratively (table driven) and other knowledge is encoded procedurally in the prototype implementation. In this effort, we took a minimalist approach to knowledge engineering, encoding as little generic, nominal knowledge as was necessary to experiment with the developing application architecture.

The "real time" report feed, however, was treated very differently in this effort. We quickly realized that it was important to obtain all the information that would realistically be available from human and automated intelligence, surveillance, and reconnaissance (ISR) reporting. For example, the initial reports that were provided to use abstracted away most of the space and time information. Target location information was limited to simple inclusion in zero or more NAIs[7] (report data that was constructed for other purposes). Such a data feed would severely limit our ability to extract collective spatial and temporal information that would be available in a real-world report setting. Therefore, we requested that the government-furnished test data be augmented with time and positional information.

Three report-feed data sets were used in this effort:

1. A manually constructed set of several dozen reports that were used during early system development work to exercise the developing software. This "quick test" data set was used until an augmented, government-supplied data set became available. The original set of "imagined" reports was eventually discarded and replaced by a sampling of reports taken from the 19Jan05 data set (below), and later, from the 08Mar05 data set.
2. The **18Jan05 data set** was provided by Christian Pizzo and contained a very large number of individual reports "observed" over a two-and-one-half hour period (15:27 to 18:00). Due to difficulties with automatically generating certain report attributes from the ground-truth sensor simulator, this data set did not contain usable direction, speed, or confidence values. This limited our ability to experiment with large-scale focus-of-attention strategies, although we did use small, manually extracted "quick test" data set that had "reasonable" values added to the extracted reports.
3. The **08Mar05 data set** was provided by Major Chester Brown and contained a large number of individual reports "observed" over a three-hour period (06:00 to 08:58). This data set contained realistic directional, speed, and confidence values, and was provided and used at the end of the effort.

The format of each report entry is shown in Table 1.

Some report attributes are retained only for informational purposes (and are not actually used in application processing). Latitude and longitude values for source and target are recorded, but only MGRS values are used for positioning. Similarly, target location within an NAI is recorded, but not used. No consistency checking of these values with the corresponding MGRS locations is performed. The current implementation also records but does not use source and target altitude values. All computations assume source and target positions are on the same horizontal plane. A report of altitude relative to the ground level at the sensed position is problematic, as conversion to a common three-dimensional reference system requires access to terrain-elevation data for each sighting. Altitude values would be more useful if they were converted to a common reference altitude (such as height above sea level (ASL)) before they are provided to the information-fusion application.

MGRS values are converted into 1-meter resolution "local Cartesian" coordinates when input reports are first ingested into the information-fusion application. This conversion enables efficient integer-based computations and takes advantage of the localized area of operation (typically involving only a few 100km square grid cells). In this initial effort, registration issues of reports from different grid cells were not addressed in performing the local-Cartesian conversion. This results in a few discontinuities in reasoning across cells.

[7]Named areas of interest, which are spatial regions where observation has been planned or is expected to be needed. In the government-supplied data used in this effort, all NAIs are circular areas.

Each report item in the data-set stream is formatted as follows:

1. **Time Data Available** — Military Date/Time Group (DDMonYY, Hour, Minute, Second, Time Zone: `30,Dec,2004,12,00,59,Z`)
2. **Time Sensed** — Military Date/Time Group (DDMonYY, Hour, Minute, Second, Time Zone: `30,Dec,2004,12,00,59,Z`)
3. **Target Type** — Enemy Battle Space Object (BSO) or aggregate according to unit identification or enemy equipment list codes: `2S3`
4. **Target Quantity** — Quantity of BSO or aggregate: `16`
5. **Target Activity** — Description of target activity using Enemy Activity Codes: `MASSG`
6. **Target Direction of Movement** — Cardinal, ordinal, or azimuth (vector) along which the target is moving (field is blank if target is stationary or source cannot provide direction information): `NW or 270`
7. **Target Speed** — Target velocity in kilometers per hour: `25`
8. **Named Area of Interest (NAI)** — NAI in which target appears if applicable (field is blank if the target is not within an NAI or the source does not provide it: `34`
9. **Target Latitude** — Location of target along the parallel of latitude (North or South, Degrees, Minutes): `N,40,42.033`
10. **Target Longitude** — Location of target along the meridian of longitude (East or West, Degrees, Minutes): `E,47,06.946`
11. **Target Altitude** — Height of target Above Ground Level (AGL) in feet: `6`
12. **Target Military Grid Reference System (MGRS) Location** — Location of target within the MGRS rectangular grid (Grid Zone Designation, 100,000 meter square identifier, 6 to 10-digit grid coordinate—100 to 1 meter accuracy respectively): `12RWV7040083640`
13. **ISR Source Platform** — Identification of the specific Sensor Platform/Source that collected and reported the information (also known as the bumper/airframe/hull/unit number or name): `2UAUAV`
14. **ISR Source Type(s)** — Identification of the types of sensors used by the ISR Source Platform to collect and report the information. If more than one type of sensor is used, the combination of sensors will be reported within parentheses separated by a space. Single source report: `SIGINT`; multiple source: `(MTI SAR)`
15. **Source Latitude** — Location of source along the parallel of latitude (North or South, Degrees, Minutes): `N,40,42.033`
16. **Source Longitude** — Location of source along the meridian of longitude (East or West, Degrees, Minutes): `E,47,06.946`
17. **Source Altitude** — Height of source Above Ground Level (AGL) in feet: `6`
18. **Source Military Grid Reference System (MGRS) Location** — Location of source within the MGRS rectangular grid (Grid Zone Designation, 100,000 meter square identifier, 6 to 10-digit grid coordinate—100 to 1 meter accuracy respectively): `12RWV7040083640`
19. **Confidence of Information** — The degree of confidence in the report, expressed as a number in the range 0–100: `80`

Table 1: Data Set Report-Item Syntax

The use of a single "confidence" value for each reports allows considerable interpretation as to the semantics of the report. One interpretation is that the confidence indicates the certainty that all provided values are within expected bounds. This is the report-semantics model used in this effort. With this interpretation, a report that says "Unknown Object" is at location x,y moving North at 30km/hr with confidence 95 means that the location, direction, and velocity are pretty certain. An otherwise identical report that classifies the object as a "Pickup Truck" means that the type identification is also pretty certain. Another report with a confidence of 50 is assumed to mean that it is equally likely that at least 1—and possibly all—of the report attributes are incorrect (outside of expected bounds) or that the report is not based on anything present in the environment[8] Clearly multiple confidence values should be developed for report data. This would provide significantly more expressiveness, allowing for a report to be very uncertain of target type, but confident about location and somewhat confident about direction and speed.

3.2 Blackboard Representations

GBBopen provides rich facilities for defining blackboard-object classes and for creating blackboard level and instance objects. In GBBopen terminology, blackboard objects are called *units* and levels are called *spaces*. Unit instances have *slots*, which are akin to fields, attributes, or instance variables in other object-based languages. GBBopen also provides special *link* slots that maintain bi-directional pointers between unit instances in a one-to-one, one-to-many, many-to-one, or many-to-many relationship. Links greatly reduce the level of bookkeeping required when instances are linked, unlinked, or deleted.

Dimensional values play a key role in GBBopen. Dimensionality provides an abstraction that separates the high-level semantics of blackboard and space instances and retrieval operations from the low-level details of repository storage and indexing machinery. Dimension values can be extracted directly from the slots of a unit instance, from computations using slot and other values, or indirectly, from unit instances linked to the unit instance.

Report data is ingested into the application in the form of `report` unit instances (Table 2). These instances contain the reporting information described in Table 1. Ingestion is performed by a stream-based reader that can accept comma-separated textual report data from a file or an on-line stream. The reader is invoked from a control-shell event function that reads the next block of available reports (based on the `data-available-time` value whenever the "world clock" needs to be updated. In the current implementation, the clock is updated on quiescence: when no executable KSAs are pending.

Aggregation processing (discussed in Section 3.4) creates `group` and `track` unit instances representing spatial and temporal aggregation, respectively (Table 3). These instances support arbitrary hierarchies of groups and tracks, supported at the bottom by individual reports.

Unit instances are also used to hold information about NAIs, source, target, and equipment types (Table 4).

Finally, `reporting-goal` unit instances are used to represent detailed spatial/temporal goals computed from individual PIRs, which are also represented by `PIR` unit instances (Table 5). High-level Bayesian reasoning objects are represented on the blackboard using unit instances that follow the AIID design [37].

KSs and KSAs are represented using GBBopen's standard KS and KSA unit instances provided by the Agenda Shell module.

3.3 Control

Control facilities in this effort were implemented using GBBopen's Agenda Shell module. The Agenda Shell provides a rich set of KS functions and predicates to manage the progression of KSAs from initial

[8]Nearly equivalent to having all the attributes wrong.

```
Unit class: report
 Slots:
  instance-name
  data-available-time
  sensed-time
  target-type
  target-quantity
  target-activity
  target-direction-of-movement
  target-direction-of-movement-uncertainty
  target-speed
  NAI
  x
  y
  target-latitude
  target-longitude
  target-altitude
  target-MGRS
  source-platform
  source-types
  source-location
  source-latitude
  source-longitude
  source-altitude
  source-MGRS
  confidence
 Links:
  tracks
  groups
 Dimensional values:
  target-type, time, x, y, confidence
```

Table 2: Report Unit Instances

triggering and activation through obviation or execution. A typical KS will only require a subset of these functions and predicates:

- An *activation-predicate* — a function that is called with two arguments, the unit instance representing the KS and the object representing the triggering event. The *activation-predicate* should return a Boolean value that indicates whether the KS should continue to be considered for activation in response to the event.
- A *precondition-function* — a function that is called with two arguments, the unit instance representing the KS and the object representing the triggering event. The *precondition-function* is called unless the *activation-predicate*, if supplied, returned nil and should return one of the following sets of values:
  - `nil` indicating the KS is not to be activated in response to the event
  - `:stop` (and, optionally, additional values to be returned by the control shell) indicating that the control shell is to exit immediately
  - An integer execution rating for the KSA (and, optionally, initialization arguments to be used when creating the KSA unit instance)
- An *execution-function* — a function that implements the KS. When an activation of the KS is executed, this function is called with one argument, the unit instance representing the KSA. If the execution function returns the value `:stop` (and, optionally, a additional values to be returned by

the control shell), the control shell will exit immediately.

- An *obviation-predicate* — a function that is called with two arguments, the unit instance representing the KSA and the object representing the obviation event. The *obviation-predicate* should return a Boolean value that indicates whether the KSA should be obviated (permanently deemed unnecessary and therefore will never be executed).
- A *retrigger-function* — a function that is called with two arguments, the unit instance representing the KSA and the object representing the retrigger event. The *retrigger-function* can perform whatever activities are needed in response to the event. Typically this involves augmenting the triggering context of the KSA or changing its execution rating.
- A *revalidation-predicate* — a function that is called with one argument, the unit instance representing the KSA. The *revalidation-predicate* is called by the control shell immediately before a KSA is executed and should return a Boolean value that indicates whether the KSA should be executed (if true) or obviated (if false).

```
Unit class: group
  Slots:
    instance-name
    group-type
    confidence
  Links:
    reports
    parent-groups
    child-groups
    parent-tracks
    child-tracks
  Dimensional values:
    group-type, time, x, y, confidence

Unit class: track
  Slots:
    instance-name
    target-type
    confidence
  Links:
    reports
    parent-tracks
    child-tracks
    parent-groups
    child-groups
  Dimensional values:
    target-type, time, x, y, confidence
```

Table 3: Group and Track Aggregate Unit Instances

The Agenda Shell provides flexible, priority-based control decisions at the KSA level: when the currently executing KSA completes, the highest rated pending KSA is selected for execution. Execution ratings are computed by the KS's precondition function (and possibly modified at a later time by the KS's retrigger function). In the information-fusion application, rating calculations involved a linear "utility" combination of the confidence in the input data for the KSA, a simple estimate of the quality and cost of the results that will be produced by the KSA (computed from the KSA trigger and associated reports based on the trigger), and the priority of any spatial/temporal reporting goals created in response to

```
Unit class: NAI
  Slots:
    instance-name
    description
    latitude
    longitude
    MGRS
    x
    y
    radius
  Dimensional values:
    x, y

Unit class: Equipment-type
  Slots:
    instance-name
    description
    function
    mobility-type
  Dimensional values:
    mobility-type

Unit class: Target-type
  Slots:
    instance-name
    description
    function
    mobility-type
  Dimensional values:
    mobility-type

Unit class: Source-platform
  Slots:
    instance-name
    description
    x
    y
    time
  Dimensional values:
    x, y, time
```

Table 4: NAI, Source, Equipment-Type, and Target-Type Unit Instances

```
Unit class: PIR
  Slots:
    instance-name
    description
    x
    y
    time
    target-types
    priority
  Links:
    reporting-goals
  Dimensional values:
    time, x, y, target-type

Unit class: reporting-goal
  Slots:
    instance-name
    x
    y
    time
    target-types
    priority
  Links:
    PIR
  Dimensional values:
    time, x, y, target-type
```

Table 5: PIR and Reporting-Goal Unit Instances

specific PIRs. The input data confidence is obtained directly from the probabilities associated with those blackboard objects. These are used to produce quality and cost estimates of the results using detailed KS-specific control knowledge that is packaged with each KS (as was shown in Figure 3). The estimated results are matched against overlapping reporting goals to obtain a set of priority values that are combined (weighted independently) to form a prioritization multiplier for the KS-specific rating value. Specifically, the rating of each KSA is:

$$rating = \text{KS-specific-rating} * \frac{\prod_{i=1}^{n} P_{g_i}}{\prod_{i=1}^{n} P_{g_i} + \prod_{i=1}^{n} (1 - P_{g_i})}$$

where P_{gi} is overlapping goal$_i$'s priority, which can range from 0–1, and 0.5 is the multiplying factor used when no overlapping goals are found for the outputs of a KSA. Thus, goal priorities greater than 0.5 increase the desirability of executing a KSA and priorities less than 0.5 tend to inhibit execution.

An important control issue with high-data-volume applications such as FBKFF is developing sufficient contextual separation to justify the cost of focus-of-attention control reasoning. For example, in the original Hearsay-II speech understanding application (the first blackboard-system application), opportunistic KSA-level control was disabled below the phrase level of the blackboard [7, 1]. It was discovered that there wasn't enough confidence information among the low-level data to predict what data should be worked on first. Instead, Hearsay-II "batch processed" all the top-rated data up to the phrase level of the blackboard, at which point opportunistic control reasoning took over. If the top data failed to produce a confident interpretation of the input data, further rounds of batch processing were invoked to add to the pool of inputs that were processed to form triggers for phrase-level processing.

FBKFF reports to have a similar character. The confidences associated with reports tend to range from 75 to 95 (percent), with many in the 90–95 range. Focusing on the highest rated reports, even biased by reporting goal priorities, does not ensure that important triggering information has not been overlooked. In addition to control-shell processing, GBBopen provides a lightweight and flexible event-function mechanism that invokes functions in response to blackboard-object creation and modifications. We used this mechanism to batch process the initial spatial and temporal aggregation of ingested reports. Conventional control-shell processing takes over on these initial semantic aggregations. Note that the issue is about focusing system activity based on the reports—not about filtering reports. Low-confidence reports remain on the blackboard to be used as needed by activities triggered by other reports. If the system isn't fast enough to look at all the low-level report data to determine what should be done (actually, by using the event-function mechanism, it's quite fast at this stage), then it is important to balance what low-level reports are used as KS triggers against system objectives that are represented by PIR goals. As with Hearsay-II, we are able to perform enough "batch" low-level processing to begin making informed higher-level control decisions.

3.4 Semantic Aggregation

Aggregating[9] individual reports in both space and time can greatly increase confidence in the identity, behavior, and intent of observed entities. In this effort, we performed basic work in automated spatial and temporal aggregation and reasoning. Associating reports of individual objects and aggregations of objects as they move in conjunction with one another over time is an important, high-level semantic activity. Such high-level "behavioral monitoring" goes far beyond kinematic object trackers in exploiting the rich, collective information that is spread among a large number of ISR sensor and human-generated reports. Unlike "tracking" sensors that continuously observe targets over time, reports to the FBKFF information-fusion application are instantaneous and disjoint in time. Many reports contain the assessed type of the referenced target, but not the identity. Thus reports of the same target type observed nearby one another in space and time may or may not be the same target as it moves through the environment.

Temporal aggregation involves associating the positions and movement of individual objects and of higher-level spatially aggregated entities over a number of observations. Temporal aggregation can greatly clarify behavioral activities that appear uncorrelated and without purpose from the perspective of individual observations at any point in time. Even at the detailed signal-processing level, very little attention has been paid to multi-sensor tracking systems with sensors that produce observations at different (asynchronous) times [44]. Associating individual reports of individual entities as they move in conjunction with one another over time is an important, high-level semantic activity that uses spatial and temporal expectations made in the context of the friendly force's mission, more global courses of action that the enemy is believed to be pursuing, the weather, and the disposition and threat capability of friendly forces.

For example, consider the observations of objects A, B, and C shown in Figure 4 that have been made at three different times. Each of the observations includes kinematics estimates of the speed and direction of each object. The actual route of each object is shown by the lightly dashed lines, but since the objects are not continuously observed, these routes are unknown to the system. Taken individually, each observation does not appear to be threatening to the friendly "Base" object. Whether they are acting for misdirection or stealth, the possible collective behavior of A, B, and C is not apparent from a single set of observations. Only when the meandering observations are considered in concert and over time are the behaviors of A, B, and C potentially threatening.

Spatial aggregation involves the application of fluid spatial-pattern knowledge that represents how individual objects operating in conjunction with one another (such as a multi-tank group) tend to be

[9] Sometimes called associating.

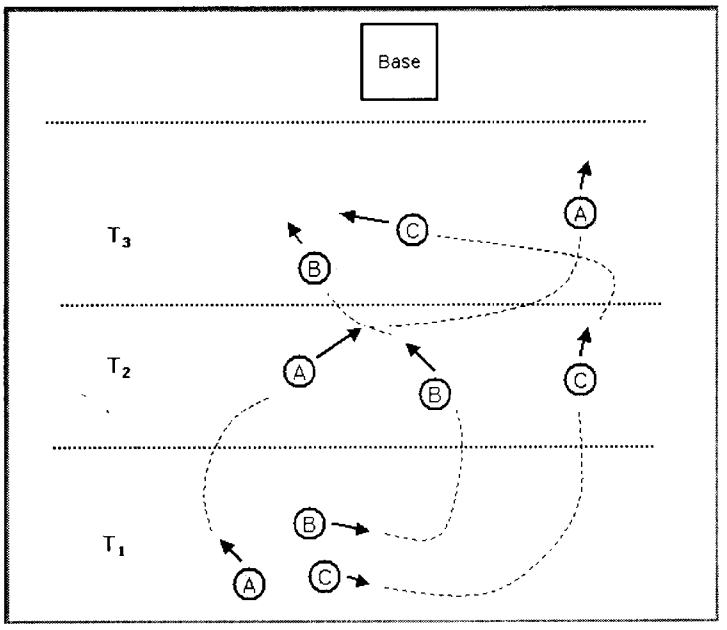


Figure 4: Temporal Aggregation

positioned and oriented. Such spatial pattern knowledge includes objects that are always expected to be operating together as well as other objects that may be, but are not necessarily, present. Expected spatial positioning can be adjusted in response to contextual terrain or weather characteristics (such as a swamp or a ridge) and to behavioral activity (such as rapid movement across open terrain).

Conventional propositional and probabilistic reasoning methods, such as ever-present Bayesian network techniques, do not handle the spatial and temporal knowledge that can be applied to the information-fusion and behavior assessment activities. Extensions, such as Dynamic Bayesian Networks (DBNs) [41] attempt to handle time-changing problems by instantiating and connecting sequences of entire Bayes networks, each representing a possible situation at a snapshot in time [45], by adding additional nodes to the graphical model representing each temporal interval of interest as a random variable [46], or by using temporal (or dynamic) probabilities to represent the changing state of the observed environment [47]. Such approaches are intractable beyond very simple settings and temporal quantifications. Recently, temporal abstraction techniques have been proposed as a means of reducing the combinatorial explosion of possibilities inherent to these temporal-reasoning approaches [48]. Analogous extensions could be used to represent uncertainty in possible spatial positioning, with similar difficulties in representation and tractability [49].

The problem with all of these reasoning extensions is that they remain, at their heart, atemporal and aspatial representations of possibilities and beliefs. The goal of maintaining all the advantages of a principled Bayesian representation and reasoning framework—well-understood properties and simple inference computations—limits the expressiveness and efficiency needed to reason with spatial and temporal knowledge in highly dynamic settings. Blackboard-system representations of complex functional knowledge and data can be used to represent temporal and spatial aggregations directly, as first class entities. Using this form of representation, time and spatial movement become explicit dimensional attributes of objects that represent aggregated behaviors—not instantaneous, atemporal observations. This representational choice allows highly complex knowledge to be represented and used, even if some of this knowledge is represented functionally rather than assertionally.

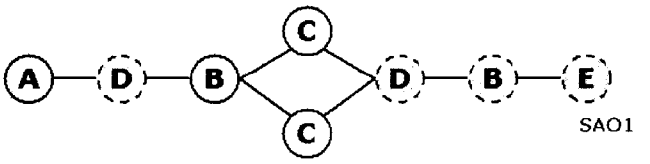


Figure 5: A Spatial-Aggregation Template

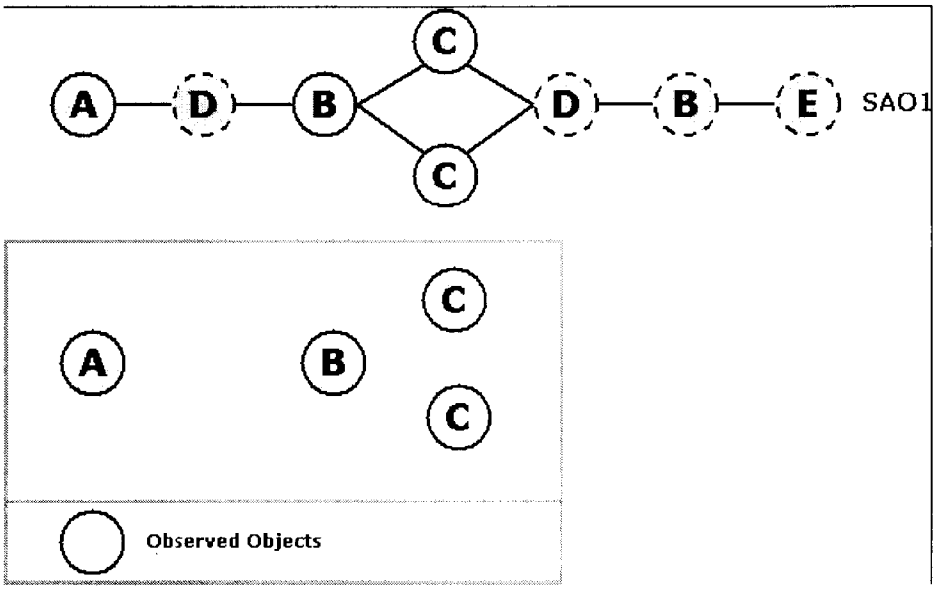


Figure 6: A Simple Template Match

Consider the following examples of the kind of knowledge and reasoning required to support collaborative information-fusion activities. Figure 5 shows the spatial-aggregation template for a pattern of objects that, when observed together, form a spatial-aggregate object (SAO). The SAO template shown is called SAO1 and represents the knowledge about how individual objects are expected to be related spatially to one another if they are members of an instance of SAO1. All instances of SAO1 should contain the objects shown in the figure as solid circles (object A, B, and two C's). Instances of SAO1 may also contain some or all of the objects shown as dotted circles (two D's, another B, and an E). All the component objects that do exist are expected to be in approximately the spatial configuration shown relative to the orientation of the SAO template. The layout of Figure 5 represents the pattern objects as if they were viewed from above, with the pattern oriented left-to-right. Not shown in the figure is functional knowledge representing the confidence attributed to SAO1 based on observing each of the expected and possible objects and how that confidence diminishes as the objects vary spatially.

To illustrate a simple SAO1 template match, Figure 6 shows the association of the expected composite objects, in their expected relative spatial positioning, and without any additional (possible) objects. The observed data contains an A object, followed by a B and two C objects in the appropriate spatial configuration. You might be viewing this aggregate object instance by looking down from an aircraft. For simplicity, the data is already aligned with the template but, in general, SAO templates will need to be rotated appropriately in order to match the composite-object orientation.

Now consider what happens if the composite objects are not observed at the same moment in time, such as if the view is partially obscured by fog or passing clouds. This is the situation that is shown in Figure 7. At a first sighting, only two cross-hatched objects are visible (A and one C). At a later time,

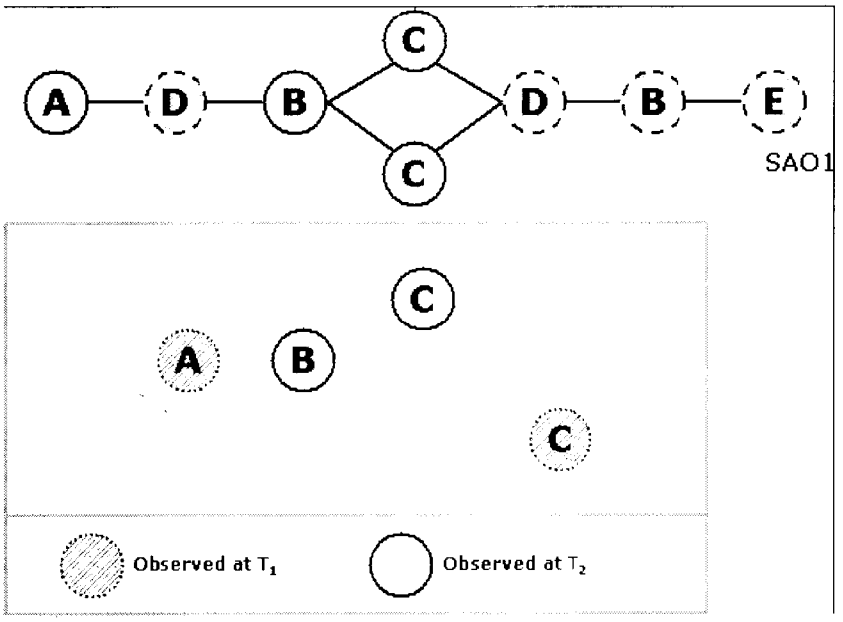


Figure 7: A Temporally Adjusted Template Match

T2, a second set of observations is obtained and this time only the two non-cross-hatched objects (B and the other C) are visible. Is this an instance of an SAO1? Without adjusting for the time of the two sets of observations, the spatial configuration of the viewed positions of the objects is not correct, but the sightings were obtained over time and the objects moved during that time. Given what is known about the direction of motion and velocity of the component object types, the components could indeed match the spatial configuration of the SAO1 template. In fact, given the initial sighting of A and C and knowledge about the estimated movement properties of SAO1 object, it is possible to predict where to expect the other objects at other times (such as time T2). As this example shows, matching SAO templates against observations at multiple times adds significant complexity to template matching. In this effort, we experimented with an approach that looked for the best match of required template objects with one or more report objects observed at the same time. Then other "nearby" objects were adjusted in space and time to this observed time (based purely on constant velocity assumptions) and matched against the template. This is a very simple interpolation approach to temporal alignment that can compensate for relatively small sensed-time differences relative to object movement. However, it is expensive and very sensitive to parameter settings of how far in time to shift and interpolate temporal data and how far in space to group related reports. There is a substantial amount of identification uncertainty in the raw reports, which requires contextual knowledge to resolve fully. For example, a aggregate of four tracked objects moving together over a period of time might require considering their behavior relative to the activities of other aggregates to select among plausible candidate interpretations. The level of knowledge engineering required for this was beyond the scope of this prototyping effort.. The availability of battlespace-object (BSO) annotation in some reports can be used to significantly reduce the space of candidates by using BSO-tracks as guides for aggregating other nearby reports. As the specific details of BSO-tracking capabilities, such as the percentage of reports that contain accurate BSO identifiers, and the degree of available sense-time synchronization among reports become better known, our initial aggregation algorithms can be significantly refined in future research work!

Figure 8a shows a number of individual object observations (made at the same time) that are very similar to an SAO1 instance. However, the data are missing one of the expected C objects and the C object that is observed is not in the expected spatial location. Therefore, confidence that these objects are part of an SAO1 instance is quite low. However, what if there are two C objects but they are lined up sequentially rather than side-by-side (as shown in Figure 8b)? Although this is closer to the SAO1

template, it is not a great match due to the spatial deviation. However, there are conditions under which the data could be considered a close match! For example, the component objects could be moving through a narrow ravine where there is not enough room for the two C objects to maintain the expected side-by-side orientation. If it can be determined that terrain or other explanations) justify the deviation from expected spatial positioning, then the formation shown in Figure 8b could still be considered a good match with SAO1. We did not attempt to address justifiable template deviations in this effort; this issue remains a future-research activity.

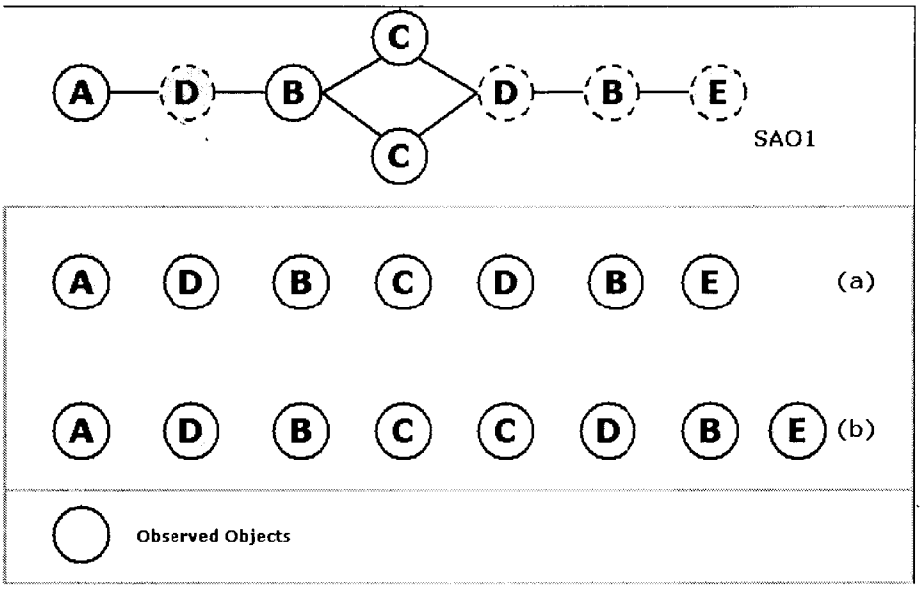


Figure 8: A Spatially Adjusted Template Match

These basic examples simplify many of the detailed aspects of spatial and temporal knowledge. For example, SAO templates are not rigid patterns with uniform confidence degradations due to differences in spatial positioning. Some aggregate patterns may involve only relative distances between composite objects, rather than directional properties. Patterns may be quite fluid, up to a point where differences suddenly make the confidence in a match very low.

Consider the complexity of applying a set of SAO templates to a large volume of individual observations. The major challenge involves orienting and adjusting the appropriate SAO templates to align with the observed data. Matches may be incomplete with respect to expected components and the data space will be cluttered with other components that can be present in the aggregate as well as many other observations that are associated with any aggregate.

Throughout the matching process, functional knowledge is needed that describes the spatial fluidity and confidence adjustments of potential template matches. The match confidence is based on the confidence values of individual components (more, highly believed components are better) and on deductions stemming from unjustified deviations from expected components. Missing, but highly expected, components may suggest where additional observations are needed in order to increase the confidence in the aggregate (template-based) object. The benefit of finding such expectations is passed to the control component as additional "power-of-information" focusing data.

Because semantic aggregation has both temporal and spatial components, the order in which aggregation is performed can significantly affect cost and complexity. Temporal aggregation reduces the uncertainty in number, position, and behavior of individual targets, greatly simplifying template matching. On the other hand, tracking aggregated groups of objects rather than individual object sightings can significantly reduce the number of temporally-aggregated tracks that need to be generated.

The report datasets that were available during this effort did not provide a sufficiently clear semantic consistency to determine temporal-first or an aggregation-first strategy performance for FBKFF. The potential availability of battlespace-object (BSO) annotation in some reports complicates the matching strategy, due to the significant reduction in candidates by using BSO-tracks as guides for aggregating other nearby reports both temporally and groupwise. As the characteristics of "realistic" FBKFF report data feeds become better known, detailed evaluation of the most effective aggregation strategies will be feasible.

3.5 User Interface

All work in this effort was performed and evaluated using the user-interaction facilities of the underlying Common Lisp implementation. At the onset, we knew that graphical interface and visualization facilities would be very important in this effort—both for assisting with the day-to-day research and development activities and for conveying to others what the prototype application is doing.

Two distinct graphical-user-interface requirements were identified. The first interface requirement is for a graphical environment suitable for Army analysts and decision makers. This display should be able to display dynamic object and threat representations on terrain maps, much as plastic transparency sheets can be used manually in conjunction with physical maps. The standard software tool that was being used for map display is FalconView,™ which was chosen as the client for this interface.

FalconView[10] is a Microsoft-Windows-based mapping system that displays various types of maps and geographically referenced overlays. It was developed by researchers at the Georgia Tech Research Institute and is available free of charge to all components of the U.S. Department of Defense. Recent FalconView development has been funded by the Air National Guard, Air Force Reserve, U.S. Special Operations Command, and Air Combat Command, and these organizations control the its distribution. Many types of maps are supported in FalconView, but the primary ones of interest to most users are aeronautical charts, satellite images and elevation maps. FalconView also supports a large number of overlay types that can be displayed over any map background. The current overlay set is targeted toward military mission-planning users and toward aviators and aviation-support personnel.
After reviewing FalconView's capabilities, we decided to pursue a dynamic overlay approach where the information-fusion application would create, modify, and delete objects on the dynamic overlay and user events, such as mouse clicks and keyboard inputs, would be returned to the application. Because FalconView is only available for Windows, we felt that a socket-based connection between the information-fusion application and a client stub attached to FalconView on the user's desktop was the most flexible architecture. Interaction bandwidth between the application and client was not expected to be an issue.

FalconView provides a number of APIs for third-party extensions. For our needs, we chose the ILayer and ICallback facilities, both part of FalconView's AutomationInterface. The ILayer interface allows separate-process applications or same-process (shared memory) COM objects to add vector based items to the FalconView map. It also allows the object to control overlay order and to open and close overlays. ICallback is a COM interface defined by FalconView but implemented by the third-party program. When the user interacts with a graphical item that was created by the third-party program, FalconView will call the appropriate method on the ICallback interface, which allows the third-party program to interact with the user.

Unfortunately, we were unable to progress beyond our basic interface design in this effort. Even with considerable assistance and persistence by Dr. Gerald Powell (U.S. Army RDECOM CERDEC I2WD), obtaining UMass access to FalconView was a protracted administrative process, and one that was not concluded until the very end of this initial research effort. Thus, implementing our design for this user interface remains a high-priority item for future work.

[10]http://www.FalconView.org/

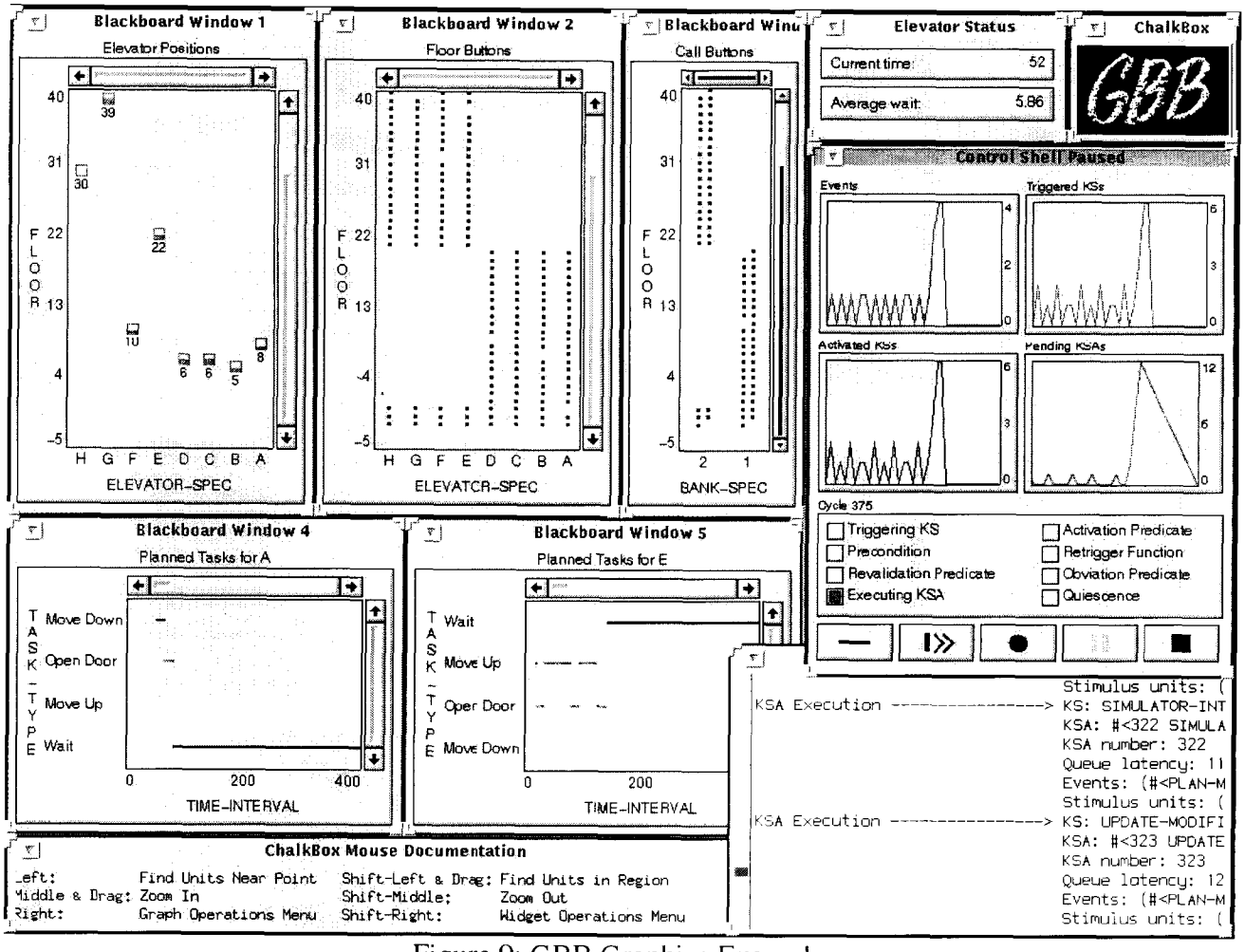


Figure 9: GBB Graphics Example

The second form of user interface would provide visualization of what is happening on the blackboard and in the control decision making of the application. This "systems level" view is tailored for understanding the details of what is going on in the information-fusion application rather than on conveying the results of that work to Army analysts and other decision makers. In this case, what is needed are interactive graphical facilities similar to those that were provided by the commercial GBB™ product (Figure 9).[11] However, open-source versions of such facilities remain to be developed by the GBBopen Project, and we did not have resources to undertake such development in this effort. As useful as these facilities would be, they remain a subject of future development.

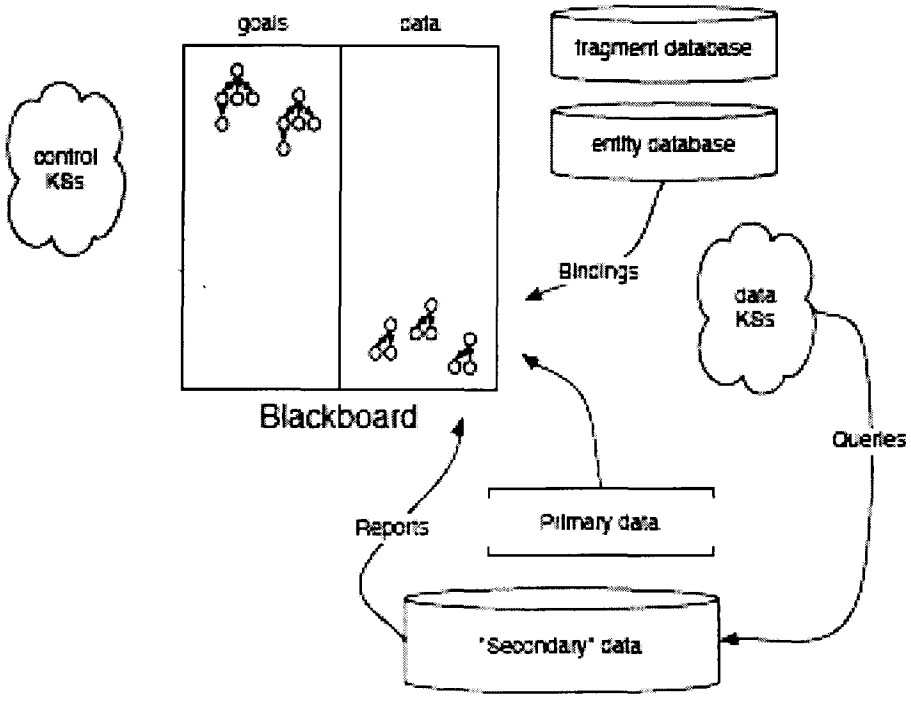


Figure 10: IJARA Architecture

3.6 Incremental, Multi-Entity Bayesian Reasoning

IJARA is the most recent implementation of EKSL's AIID Bayesian-blackboard approach [36] and builds on the ideas of Laskey, et al [33, 50]. IJARA was used for the high-level Bayesian reasoning exploration in this effort.

Following the goal-directed blackboard architecture design of Corkill and Lesser [51, 52], IJARA's blackboard is divided into two sides: one for goals and one for data. Control KSs that are associated with goals operate on one side and data KSs that are associated with data operate on the other side (Figure 10). Data arrives on the IJARA blackboard as report instances. IJARA report instances may come from human or sensor reports (primary data) or as a result of user-generated or blackboard queries (secondary data). Once report instances are on the blackboard, two types of processing occur:

- domain-dependent data KSs look for patterns in the primary data
- domain-dependent goal KSs attempt to satisfy goals (secondary data) that are posted to the goal side of the blackboard

These two activities interact via other generic KSs that attempt to connect goals to data in order to build complete Bayesian networks (BNs). As much as possible, data on the blackboard is represented by random variables (RVs) and Bayesian fragments (BFs). These domain-specific fragments are part of the knowledge engineering required to use IJARA. IJARA creates hypotheses as fragments are added to the blackboard and bound to specific entities. These hypotheses form a forest of search trees and are used to focus processing attention. Structurally equivalent BFs and RVs can appear multiple times on the blackboard with their arguments bound to different entities.

[11] As the result of a sequence of corporate acquisitions, the GBB product software is no longer available or supported.

Report processing IJARA has two canonical report types: reports about entity attributes and reports about link attributes. Entity-attribute reports say that some attribute of some entity has some value with some belief. Link-attribute reports say that some attribute of some link between two entities has some value with some belief. Both report types introduce named entities to the blackboard. These names are used to link reports with any RVs or BFs that pertain to them.

When a raw report unit appears on the blackboard it triggers a generic report triage KS. This KS:

- finds or creates any entities or links mentioned by the report
- connects the entities and links to the report
- stores the report for later processing

Domain-specific KSs are triggered if anything interesting has happened on the blackboard. If something has, they will post bound RVs or BFs to the blackboard. Most reports will not trigger any further processing. These will eventually be removed from the blackboard and placed in long-term secondary storage. The current IJARA system does not implement this feature.

Some reports result from queries by the user or by IJARA. These are connected to the query that caused their generation. Query processing is quite complex because IJARA must monitor each query to ensure that it has completed in a timely manner. Queries that fail can be given more time or resources and allowed to continue. This more complex query handling is domain specific and is not implemented in the current version of IJARA.

Blackboard processing In operation, IJARA repeatedly selects one of the following operations and executes it. The specific order of operations is opportunistic, and the search through hypothesis and binding spaces can be driven by value-of-information reasoning, entropy minimization, or other, possibly ad hoc, reasoning methods

- High-level goals are posted to the blackboard. These are represented as a desire to know the value of some RV with some confidence. For example, is there suspicious enemy activity in NAI 34?
- If the RV's value cannot be determined by direct observation, a generic KS fires in an effort to find BFs that can be used as additional supports for the RV or extend its use. A hypothesis object is created for each possible extension.
- These hypotheses will typically have unbound attributes. Other generic KSs will fire in an effort to find and create bindings for these attributes. Domain-specific KSs can be used to help in prioritizing binding decisions. The hypotheses will be extended for each binding created.
- A generic KS is used to find existing BFs that can be extended by combining them with other fragments from the fragment database.
- The execution of domain-specific KSs may query existing BNs, search for additional data, or execute algorithms on the data already on the blackboard. These queries and algorithms may write additional BFs on the blackboard or alter existing ones.

4 Lessons Learned

The use of existing blackboard software technology as the foundation for the architectural thrust of this effort significantly reduced the technical risk in creating the initial collaborative FBKFF prototype. Nevertheless, this work is pushing the frontier of temporal and spatial aggregation techniques, principled result integration, and dynamic power-of-information control approaches. The research performed in this effort is an important first step to meeting the challenge of an effective collaborative software application for information fusion.

What have we learned from this effort? First, using a blackboard-system architecture and GBBopen was a good choice for this effort. The architectural flexibility allowed us to make many strategic representation and control changes as this work progressed, and we expect continued adaptation and extension will be important in future research. The architecture allowed us to work with low-level report

processing and semantic aggregation and with the AIID/IJARA approach in the same environment. The programming productivity provided by GBBopen and its Common Lisp foundation allowed us to experiment and change our representations and computational methods quickly and easily.

We also learned early on that it is important to include all the information that can be obtained from human and automated ISR reporting. In this effort, we often simplified our processing of this information for initial expediency, but the report information was available when we could make use of it. In research such as this, up-stream simplification of report data for downstream simplicity is ill advised—it is very difficult to predict what information may be highly beneficial to information-fusion processing. For example, abstracting away most of the space and time information from reports (to simple inclusion in NAIs) results in a data feed that is severely limited. We strongly recommend that all realistic report data be made accessible to information-fusion-application research. Our sponsor has indicated that generating realistic report data of the type desired and being able to tie it to scenarios is a limitation of the current modeling and simulation environments.

We hypothesized that it would not be difficult to migrate AIID research to the GBBopen-based application environment. While this implementational assertion proved to be true, we also discovered that a significant disconnect remains between the large-volume processing of low-level reports and the higher-level semantic reasoning that can be performed using IJARA. It remains unclear that the AIID/IJARA prototype, as it is now implemented, can be scaled to the spatial and temporal complexity, and to the sheer volume, required for the FBKFF application. Not only is the AIID/IJARA prototype not very far along in design and implementation, it does not focus on the key issues of more principled integration of KS contributions or highly spatial and temporal aggregation. This observation does not imply that significant advances in principled integration and aggregation reasoning are not possible, only that they are not well supported in the AIID/IJARA prototype. A new, formal contribution-integration design that supports more effective principled, incremental Bayesian reasoning appears preferable to evolving the existing AIID/IJARA implementation and is an important component of future research.

The adjective "principled" has become a nearly content-free (and almost religious) label that is being routinely used to distinguish "good" AI representation and inferencing approaches from "bad" AI (typically historic, ad hoc, techniques). It is also invoked to claim that one approach is "more principled" than another. Often "principled" is used to indicate use of a graphical-network (Bayesian or probabilistic) representation, as if the mere use of an approach was sufficient to be "good," reasonable, or sound. No one proclaims that their techniques are "un-principled," but the techniques in and of themselves do not quantify the soundness of an approach or application.

For example, one might ask if current Level-1 fusion techniques that integrate multiple observations of multiple instances of a single-source type, such as Ground Moving Target Indicators (GMTI), or from multiple observations of multiple instances of multiple source types (GMTI and Infrared), are reasonably "principled?" Many Level-1 approaches use training data or some formal underpinning in relating multiple observations together. ("If I see X enough times, I'm pretty sure it's really there and really X.") However, the issues of what constitutes a probabilistic observation assessment (versus a confidence in an observation) involves the detailed semantics of the sensory data, how they are combined (from the same sensors or different sensors), the processing that is performed on them (corroborating versus redundant), and the state of the environment (possible confusion with other objects, near-continuous versus occasional observability, temporal separation of observations, and so on).

No matter what representation techniques are used, drawing an arbitrary line in the sand and categorizing approaches as being on one side ("principled") or the other ("bad") is not helpful in assessing the effectiveness and trustworthiness of an application. A much closer look at what is going on is required to assess the system's reasonableness, soundness, accuracy, etc. Increasing the formal soundness of fusion (and collaborating-software) systems is a goal we must continue to work toward, but we must not be lulled into thinking we have achieved it solely by the use of a particular representation or reasoning technique.

5 Remaining Technical Issues & Recommendations for Future Work

Although promising, our initial effort represents only a first step in achieving an effective information fusion support application. The following technical issues and research challenges are important next steps in moving this initial effort forward:

- Further development and exploration of blackboard-based temporal and spatial aggregation and abstraction approaches. This includes addressing the limitations discussed in Section 3.4 such as incorporating justifiable template deviations and more efficient and taskable aggregation techniques.
- Research into the principled integration of sensor data, human-generated reports, and automated processing results. Instead of focusing on a principled representation of the solution, such as most of the work toward Bayesian-blackboard systems has done, the emphasis should be on making the integration of the information, work product, and results contributed by diverse reporting and problem-solving entities well founded.
- Development of dynamic, power-of-information-based, control strategies that are responsive to the current environmental context and PIRs. Principled domain and result-integration models can be used to determine the effect that obtaining particular observations or performing particular processing activities will have on the current assessment of the environment. Such power-of-information and power-of-reasoning control decisions are very useful in making effective use of sensing and processing resources.
- Exploration of graphical user-interface techniques that lead to an productive partnership between Army analysts and decision makers and the automated information-fusion application. The effectiveness of the user-interface is an important aspect of achieving efficient human-system collaboration. We recommend that the user interface designs discussed in this report be pursued in any follow-on effort.
- Increasing the scope and depth/realism of fusion and assessment knowledge (knowledge sources) used in the support application. Such detailed knowledge engineering is important in better assessing how the information-fusion application performs in larger and more realistic scenarios. Some of this assessment can be done using generic, nominal knowledge of battlespace entities, typical behaviors, doctrine, and strategies. However, we also recommend that elicitation and implementation of detailed, Army-specific and operationally accurate analysis, interpretation, control, and decision-making be undertaken (outside the public-university setting that was home to this initial research effort).
- Demonstrating and evaluating the effectiveness of the support application as it scales to large, complex, and asymmetric scenarios. We did not complete an end-to-end, reports to assessment demonstration of the information-fusion application in this initial effort. Although we made significant progress in using the collective information present in the report stream, we did not "close the loop" in terms of working collaboratively with human analysts in supporting their activities. Particularly important is expressing the PIR-based objectives to the system. Such end-to-end, full-system demonstration and evaluation remain to be done and will serve to measure the ultimate success of this initial research effort.

Acknowledgments

Dr. Gerald M. Powell (U.S. Army RDECOM CERDEC I2WD) and Major Chester F. Brown (USAIC&FH) provided many hours of background information and question answering. Their patience and enthusiasm for this work were important contributors to the progress we made in this effort. Christian Pizzo generated the initial very large (18Jan05) expanded report log. Chet Brown generated the large (08Mar05) manually annotated report log. Dr. Powell also provided many useful comments on early drafts of this report.

References

[1] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, **12**(2): 213–253, June 1980.

[2] Daniel D. Corkill. Blackboard systems. *AI Expert*, **6**(9):40–47, September 1991.

[3] Robert S. Engelmore and Anthony Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

[4] V. Jagannathan, Rajendra Dodhiawala, and Lawrence S. Baum, editors. *Blackboard Architectures and Applications*. Academic Press, 1989.

[5] Mary Shaw and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[6] Mary Shaw and Paul Clements. A field guide to boxology: Preliminary classification of architectural styles for software systems. In *COMPSAC '97 International Computer Software and Applications Conference*, pages 6–13, August 1997.

[7] Victor R. Lesser and Lee D. Erman. A retrospective view of the Hearsay-II architecture. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 790–800, Cambridge, Massachusetts, August 1977.

[8] H. Penny Nii, Edward A. Feigenbaum, John J. Anton, and A. J. Rockmore. Signal-to-symbol transformation: HASP/SIAP case study. *AI Magazine*, **3**(2):23–35, Spring 1982.

[9] Daniel D. Corkill, Kevin Q. Gallagher, and Philip M. Johnson. Achieving flexibility, efficiency, and generality in blackboard architectures. In *Proceedings of the National Conference on Artificial Intelligence*, pages 18–23, Seattle, Washington, July 1987. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 451–456, Morgan Kaufmann, 1988.)

[10] Daniel D. Corkill and Kevin Q. Gallagher. Tuning a blackboard-based application: A case study using GBB. In *Proceedings of the National Conference on Artificial Intelligence*, pages 671–676, St. Paul, Minnesota, August 1988.

[11] Kevin Q. Gallagher and Daniel D. Corkill. Performance aspects of GBB. In V. Jagannathan, Rajendra Dodhiawala, and Lawrence S. Baum, editors, *Blackboard Architectures and Applications*, pages 323–346. Academic Press, 1989.

[12] Eric Freeman, Susanne Hupfer, and Ken Arnold. *JavaSpaces™ Principles, Patterns, and Practice*. Addison-Wesley, 1999.

[13] Peter Wyckoff. TSpaces. *IBM Systems Journal*, **37**(3), August 1998.

[14] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, **32**(4):444–458, 1989.

[15] Daniel D. Corkill, October 2003. Invited talk remarks, International Lisp Conference, New York, New York.

[16] David Keil and Dina Goldin. Modeling indirect interaction in open computational systems. In *Proceedings of the First International Workshop on Theory and Practice of Open Computational Systems* (TAPOCS'03), Linz, Austria, IEEE Computer Society Press, June 2003.

[17] Frederick Hayes-Roth and Victor R. Lesser. Focus of attention in the Hearsay-II system. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 27–35, Cambridge, Massachusetts, August 1977.

[18] Richard D. Fennell and Victor R. Lesser. Parallelism in Artificial Intelligence problem solving: A case study of Hearsay II. *IEEE Transactions on Computers*, **C-26**(2): 98–111, February 1977. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 106-119, Morgan Kaufmann, 1988.)

[19] Victor R. Lesser and Lee D. Erman. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, **C-29**(12):1144–1163, December 1980. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 120–139, Morgan Kaufmann, 1988.)

[20] Victor R. Lesser and Daniel D. Corkill. The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem-solving networks. *AI Magazine*, **4**(3):15–33, Fall 1983. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 353–386, Addison-Wesley, 1988 and in *Readings from AI Magazine: Volumes 1–5*, Robert Engelmore, editor, pages 69–85, AAAI, Menlo Park, California, 1988.)

[21] Daniel D. Corkill and Victor R. Lesser. The use of meta-level control for coordination in a distributed problem-solving network. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, Karlsruhe, Federal Republic of Germany, August 1983. (Also published *in Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li, editors, IEEE Computer Society Press, pages 507–515, 1986.)

[22] Keith Decker, Alan Garvey, Marty Humphrey, and Victor Lesser. Effects of parallelism on blackboard system scheduling. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 15–21, Sydney, Australia, August 1991.

[23] Edmund H. Durfee and Victor R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-21**(5):1167–1183, May 1991.

[24] Norman Carver and Victor R. Lesser. The evolution of blackboard control architectures. *Expert Systems with Applications*, special issue on The Blackboard Paradigm and Its Applications, 7(1):1–30, 1994.

[25] Victor R. Lesser. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1:89–111, 1998.

[26] V. R. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. Nagendra Prasad, A. Raja, R. Vincent, P. Xuan, and X. Q. Zhang. Evolution of the GPGP/TÆMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems*, **9**(1):87–143, July 2004.

[27] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[28] Judea Pearl. Causality: Models, *Reasoning and Inference*. Cambridge University Press, 2000.

[29] Spirtes, Glymour, and Scheines. *Causation, Prediction and Search*. MIT Press, 2nd edition, 2001.

[30] Bill Shipley. *Cause and Correlation in Biology*. Cambridge University Press, 2000.

[31] Glymour and Cooper, editors. *Computation, Causation and Discovery*. MIT Press, 1999.

[32] Robert P. Goldman and Eugene Charniak. A language for construction of belief networks. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, **15**(3):196–208, 1993.

[33] Kathryn Blackmond Laskey and Suzanne M. Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence* (UAI-97), pages 302–313. Morgan Kaufman, 1997.

[34] Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference* (UAI-97), pages 302–313, San Francisco, California, 1997.

[35] Avi Pfeffer, Daphne Koller, Brian Milch, and Ken T Takusagawa. SPOOK: A system for probabilistic object-oriented knowledge representation. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in AI* (UAI-99), pages 541–550. Morgan Kaufman, 1999.

[36] Charles A. Sutton, Brendan Burns, Clayton T. Morrison, and Paul R. Cohen. Guided incremental construction of belief networks. In Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt, editors, *Advances in Intelligent Data Analysis V*, 5th International Symposium on Intelligent Data Analysis (IDA 2003), volume 2810 of Lecture Notes in Computer Science, pages 533–543, Berlin: Springer. August 2003.

[37] Charles Sutton, Clayton Morrison, Paul R. Cohen, Joshua Moody, and Jafar Abidi. A Bayesian blackboard for information fusion. In *Proceedings of the Seventh International Conference on Information Fusion* (Fusion 2004), pages 1111–1116, Stockholm, Sweden, June 2004.

[38] Kathryn Blackmond Laskey. First-order Bayesian logic. Technical report, George Mason University, Department of Systems Engineering and Operations Research, April 2005.

[39] Joseph Y. Halpern. An analysis of first-order logics of probability. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1375–1381, Detroit, Michigan, August 1989.

[40] Kristian Kersting and Luc De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.

[41] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference, and Learning*. PhD thesis, University of California at Berkeley, Berkeley, California, July 2002.

[42] Daniel D. Corkill, Kevin Q. Gallagher, and Kelly E. Murray. GBB: A generic blackboard development system.

In *Proceedings of the National Conference on Artificial Intelligence*, pages 1008–1014, Philadelphia, Pennsylvania, August 1986. (Also published in *Blackboard Systems*, Robert S. Engelmore and Anthony Morgan, editors, pages 503–518, Addison-Wesley, 1988.).

[43] Daniel D. Corkill. Countdown to success: Dynamic objects, GBB, and RADARSAT-1. *Communications of the ACM*, **40**(5):848–858, May 1997.

[44] Ruixin Niu, Pramod Varshney, Kishan Mehrotra, and Chilukuri Mohan. Temporal fusion in multi-sensor target tracking systems. In *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, pages 1030–1037, Annapolis, Maryland, July 2002.

[45] Thomas Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, **5**(3):142–150, August 1989.

[46] Carlo Berzuini. Representing time in causal probabilistic networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 15–28, 1989.

[47] A. Y. Tawfik and E. M. Neufeld. Temporal reasoning and Bayesian networks. *Computational Intelligence*, **16**(3):349–377, August 2000.

[48] Brendan Burns and Clayton T. Morrison. Temporal abstraction in Bayesian networks. In *Working Notes of the AAAI Spring Symposium Workshop: Foundation and Applications of Spatio-Temporal Reasoning* (FASTR), Palo Alto, California, 2003.

[49] Julien Diard, Pierre Bessière, and Emmanuel Mazer. Hierarchies of probabilistic models of navigation: The Bayesian map and the abstraction operator. In *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA04), pages 3837–3842, New Orleans, Louisiana, 2004.

[50] Kathryn Blackmond Laskey. Knowledge representation and model construction, Spring 2000. Class notes, George Mason University.

[51] Daniel D. Corkill, Victor R. Lesser, and Eva Hudlickà. Unifying data-directed and goal-directed control: An example and experiments. In *Proceedings of the National Conference on Artificial Intelligence*, pages 143–147, Pittsburgh, Pennsylvania, August 1982.

[52] Victor R. Lesser, Daniel D. Corkill, Joseph A. Hernandez, and Robert C. Whitehair. Focus of control through goal relationships. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 497–503, Detroit, Michigan, August 1989.

A Installing and Running the Prototype

Before installing the FBKFF architectural-development prototype, install a supported Common Lisp implementation and the GBBopen code. The latest Subversion (SVN) snapshot archive can be found in the "Downloads" area of the GBBopen web site (`http://GBBopen.org/`). The on-line GBBopen hyperdoc reference and the GBBopen Reference Manual (in PDF format) can be found in the "Documentation" area of the web site. Links to supported Common Lisp implementations can be found on the "Current ports" page of the GBBopen web site. GBBopen includes a mini-module system (a very simple system definition facility) that supports compiling and loading GBBopen modules. To compile all the core GBBopen modules and a simple "trip test" file, evaluate the following within your Common Lisp environment:

```
> (load "<install-directory>startup.lisp")
> (mini-module:compile-module :gbbopen-test :propagate :create-dirs)
```

GBBopen should compile, load, and run the trip test without error. The prototype application also uses GBBopen's Agenda Shell module. After exiting and restarting a fresh Common Lisp environment, evaluate the following:

```
> (load "<install-directory>startup.lisp")
> (mini-module:compile-module :agenda-shell-test :propagate :create-dirs)
```

Again, the Agenda Shell and test should run without errors. Next, extract the files from the prototype-application archive into a directory of your choice (we will refer to this directory as the "application-install-directory"). Then, create a `gbbopen-init.lisp` file your "home" directory (as defined by the Common Lisp function **user-homedir-pathname**) containing the following forms:

```
(in-package :cl-user)
(load "<application-install-directory>modules")
```

This personal GBBopen initialization file is automatically loaded by GBBopen at the end of loading the `<install-directory>startup.lisp` file, and loading it will define the prototype FBKFF application modules. To compile the FBKFF prototype-application modules, evaluate the following within your Common Lisp environment:

```
> (load "<install-directory>startup.lisp")
> (mini-module:compile-module :FBKFF :propagate :create-dirs)
```

The FBKFF architectural-development prototype should compile and load without errors, and place your Common Lisp listener in the `:FBKFF` package. To run a short, simple test, evaluate the form `(ff)`, which uses the "quick test" data set. The form `(ff8)`, runs the **08Mar05 data set**. These functions will ingest the data set and perform preliminary aggregation searches on the reports as they stream into the prototype application.